Algorithms
Computer Science 140 & Mathematics 168
Spring 2007
Homework 8a
Due Thursday, March 22

1. **[15 Points] Arbitrage: An Algorithmic Get-Rich-Quick Scheme!** You've been hired back by the Wall Street firm of Weil, Proffet, and Howe at a whopping salary. (We won't be specific about the actual amount, but let's just say that your salary is most easily expressed using scientific notation!)

   Weil, Proffet, and Howe has just entered the **arbitrage** business. Arbitrage is a money-making scheme involving anomalies in international currency exchange rates. For example, imagine that 1 U.S. dollar buys 0.8 Zambian kwachas, 1 Zambian kwacha buys 10 Mongolian tughriks, and 1 Mongolian tughrik buys 0.15 U.S. dollars. Then, by converting currencies, a trader can start with 1 U.S. dollar and buy $0.8 \times 10 \times 0.15 = 1.2$ U.S. dollars. By capitalizing on such anomalies quickly (before they're detected and corrected by the markets), huge amounts of money can be made. This, of course, requires very fast algorithms![1]

   We'll assume that we're given n currencies and the exchange rate between every pair of currencies. That is, we are given currencies $c_1, \ldots, c_n$ and some structure that allows us to determine that one unit of currency c_i buys $R[i, j]$ units of currency c_j. (You'll get to choose the data structure that stores this conversion information.)

   (a) For this first part of the problem, assume that there exist no cycles that allow you to get arbitrarily rich via arbitrage. Here's your objective: Given the list of currencies and exchange rates and a single particular currency c_i, determine the maximum amount of each currency that you can obtain, beginning with 1 unit of currency c_i. Describe an algorithm for solving this problem, briefly explain why it is correct, and give its running time. You should specify the data structure that your algorithm expects to receive containing the currency exchange rates. For full credit, make sure

[1] It's interesting to note that many algorithms experts do indeed work on Wall Street!

that your algorithm is as fast as possible. (There are several known solutions that are equally fast.)

(b) Now assume that the exchange rates are such that it may be possible to get rich via arbitrage. That is, there may exist a cycle of currency exchanges that allows you to make more of your starting currency than you had initially. Describe how you could modify your algorithm to determine if it's possible to profit from arbitrage with the given currencies and exchange rates. (Your algorithm need only return a boolean indicating whether or not it is possible to make profit using arbitrage - although augmenting the algorithm to find an actual profitable cycle if one exists is not difficult and its description is worth *3 bonus points*.) Briefly sketch the proof the correctness of your algorithm and give its running time. For full credit, make sure that your algorithm is as fast as possible. (There are several known solutions that are equally fast.)

2. **[20 Points] Napquest Re-revisited!** Napquest provides its users with the ability to find a shortest path from a starting point to a destination point. Because the graphs (maps) in general contain cycles but all of the edge weights are positive, they currently use the standard implementation of Dijkstra's Algorithm to find shortest paths. Recall that our implementation of Dijkstra's Algorithm used a heap and had running time of $O(m \log n)$.

   Napquest realizes that they need to provide the fastest possible response time to their clients and they would therefore like to improve the efficiency of their implementation of Dijkstra's Algorithm. They notice that all of the edge weights in their maps are integers in the range from 1 to W, where W is some integer constant. They believe that this extra constraint on the edge weights may be exploited to achieve an even faster implementation of Dijkstra's Algorithm. This is where you come in!

   (a) Under the assumption that all edge weights are positive integers in the range from 1 to W, describe an implementation of Dijkstra's Algorithm to compute the shortest path lengths from a source vertex to all other vertices in the graph in time $O(Wn + m)$.

   (b) Now, we'll do even better! Again under the assumption that all

edge weights are positive integers in the range from 1 to W, describe an implementation of Dijkstra's Algorithm to compute the shortest path lengths from a source vertex to all other vertices in the graph in time $O((m+n)\log W)$. (*Note:* You may find it useful to observe - and prove - that the possible set of distinct values for the labels of the unlocked vertices is relatively small. Then, you can use an appropriate data structure to get your desired running time.)